AF/2153

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE
BEFORE THE BOARD OF PATENT APPEALS AND INTERFERENCES

In re Application of:

Saulpaugh, et al.

Serial No. 09/672,145

Filed: September 27, 2000

For: Remote Method Invocation with Secure Messaging in a Distributed Computing Environment

§ Group Art Unit: 2153
§
§ Examiner: Brancolini, John R.
§
§ Atty. Dkt. No.: 5181-67300
§ P4970

CERTIFICATE OF MAILING
37 C.F.R. § 1.8

I hereby certify that this correspondence is being deposited with the U.S. Postal Service with sufficient postage as First Class Mail in an envelope addressed to: Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450, on the date indicated below:

Robert C. Kowert
Name of Registered Representative

December 1, 2004 _____
Date Signature

APPEAL BRIEF

Mail Stop Appeal Brief - Patents
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

Sir/Madam:

Further to the Notice of Appeal filed October 4, 2004, Appellants present this Appeal Brief. Appellants respectfully request that this appeal be considered by the Board of Patent Appeals and Interferences.

I. REAL PARTY IN INTEREST

The subject application is owned by Sun Microsystems, Inc., a corporation organized and existing under and by virtue of the laws of the State of Delaware, and having its principal place of business at 4150 Network Circle, Santa Clara, CA 95054, as evidenced by the assignment recorded at Reel 011464, Frame 0243.

II. RELATED APPEALS AND INTERFERENCES

No other appeals, interferences or judicial proceedings are known which would be related to, directly affect or be directly affected by or have a bearing on the Board's decision in this appeal.

III. STATUS OF CLAIMS

Claims 1-5, 11-28, 34-54, 59 and 63-68 stand rejected.

Claims 6-10, 29-33 and 60-62 are objected to.

Claims 55-58 have been allowed.

The rejection of claims 1-5, 11-28, 34-54, 59 and 63-68 is being appealed. A copy of claims 1-5, 11-28, 34-54, 59 and 63-68 is included in the Claims Appendix hereto.

IV. STATUS OF AMENDMENTS

No amendments to the claims have been submitted subsequent to the final rejection.

V. SUMMARY OF CLAIMED SUBJECT MATTER

Independent claim 1 is directed to a method for remotely invoking a method in a distributed environment in which a client generates a message in a data representation

language. The message includes a representation of a computer programming language method call. The message also includes an access credential. FIG. 6 illustrates one embodiment where a client 110 exchanges messages in a data representation language, e.g. XML in this embodiment, with services 112. Other embodiments are illustrated at figures 11b, 14, 34, 35a, and 35b, and are described at page 26, lines 18-30, page 35, line 28-page 36, line 16, page 54, line 7-page 60, line 26. Such a message may include information representing a computer programming language method call as illustrated in FIGs. 14 and 45a and described at pages 58-60. The message may also include a credential for allowing the client access to a service that is configured to perform functions on behalf of clients in the distributed computing environment as illustrated in figures 42c and 45a and described on page 60, lines 14-26. (*See also*, page 15, lines 23-30, page 16, lines 7-12, page 59, line24 – page 60, line 4, and page 61, lines 9-20).

The client sends the generated data representation language message to the service and the service examines the credential included in the message to determine the authenticity of the credential. If the credential is authentic, the service performs the requested function on the behalf of the client in accordance with the information representing the programming language method call included in the message. Alternatively, if the credential is not authentic, the service does not perform the function on behalf of the client. (*See* page 38, lines 5-12, page 59, lines 1-12, page 61, lines 9-20, page 102, lines 23-30, page 108, lines 14-21, page 110, line 20 - page 112, line 30, and "Authentication and Security" section - pages 104-136).

Independent claim 25 is directed to a distributed computing system including a service device and a client device (*See* figures 6, 11b, 14, 34, 35a and 35b). The service device includes one or more functions that may be executed on behalf of client devices in the distributed computing system as described on page 15, lines 23-30 and page 59, line 24 through page 60, line 4. The client device may be configured to generate a message in a data representation language. For instance, FIG. 6 illustrates on embodiment where a client 110 exchanges messages in a data representation language (e.g. XML in this

embodiment) with a service 112. (*See also*, 11b, 14, 34, 35a, 35b, page 26, lines 18-30, page 35, line 28-page 36, line 16, page 58, lines 1-13, page 60, lines 14-26). Such a data representation language message may includes information representing a computer programming language method call as illustrated at FIG. 45a, and described at page 58, lines 1-13. Such a message may also include a credential (FIG. 42c, page 25, lines 1-3, page 38, lines 5-12, page 61, lines 9-20, page 104, line 18 – page 136, line 11) for allowing the client device access to the service device (FIG. 45a, page 59, line24 – page 60, line 4). The client device sends the message to the service device. The service device examines the credential in the message to verify its authenticity. If the service device successfully verifies the credential, the service device performs, on behalf of the client device, a function according to the information representing the computer programming language method call included in the message. If, however, the service device does not verify the credential, it does not perform the function on behalf of the client. (*See* page 38, lines 5-12, page 59, lines 1-12, page 61, lines 9-20, page 102, lines 23-30, page 108, lines 14-21, and "Authentication and Security" section - pages 110-112).

Independent claim 45 is directed to a device including a client component and a method gate. For an example embodiment including a method gate refer to figure 14 and page 13, lines 4-18 (*See also*, "Method Gates" section – pages 32 – 64). The client component of the device may be configured to generate a computer programming language method call and the method gate may be configured to access that computer programming language method call and to generate a message. For instance, at page 58, lines 15-29 the specification describes an embodiment wherein a method gate may access a method call generated by a client process. The generated message may include information representing the computer programming language method call and a credential for allowing the client component access to a service in a distributed computing environment. The method gate may generate the message in a data representation language and may send the message to the service in the distributed computing environment. The service is operable to verify the message's authenticity by examining the included credential. If the message is verified as authentic, the service is

operable to perform a function on behalf of the client component according to the information in the message representing the computer programming language method call (*See e.g.* page 38, lines 5-12, page 59, lines 1-12, page 61, lines 9-20, page 102, lines 23-30, page 108, lines 14-21, and "Authentication and Security section - pages 110-112).

Independent claim 49 is directed to a device including a client component and a message endpoint (Figs 10b, 11a, 14, page 35, lines 17-26, page 38, line 22 – page 39, line 4). The client component is configured to generate a message that includes information representing a computer programming language method call (FIG. 45a, page 15, lines 18-21, page 58, lines 1-13). The message may be generated in a data representation language (FIGs 6, 11b, 14, 34, 35a, 35b, page 26, lines 18-30, page 35, line 28-page 36, line 16, page 58, lines 1-13, page 60, lines 14-26). The message endpoint is configured to attach to the message a credential allowing the client component access to a service in a distributed computing environment and to send the message to the service in the distributed computing environment (page 38, lines 5-12). The service is operable to verify the authenticity of the message by examining the included credential and, if the message if verified as authentic, to perform a function on behalf of the client component in accordance with the information in the message representing the computer programming language method call (*See* page 38, lines 5-12, page 59, lines 1-12, page 61, lines 9-20, page 102, lines 23-30, page 108, lines 14-21, and "Authentication and Security section - pages 110-112).

Independent claim 59 is directed to a medium (page 190, lines 1-7) including program instruction executable by a computer to implement a client that generates a message in a data representation language (FIGs. 6, 11b, 14, 34, 35a, 35b, page 26, lines 18-30, page 35, line 28-page 36, line 16, page 58, lines 1-13, page 60, lines 14-26). The message includes information representing a computer programming language method call (FIG. 45a, page 58, lines 1-13) and also includes a credential allowing the client access to a service configured to perform functions on behalf of client in a distributed computing environment (FIG. 42c, page 25, lines 1-3, page 38, lines 5-12, page 61, lines

9-20, page 104, line 18 – page 136, line 11). The client sends the message to the service and the service examines the included credential. If the credential is authentic, the service performs a function on behalf of the client according to the information in the message representing the computer programming language method call included in the message and does not perform the function if the credential is not authentic (*See* page 38, lines 5-12, page 59, lines 1-12, page 61, lines 9-20, page 102, lines 23-30, page 108, lines 14-21, page 110, line 20 - page 112, line 30).

VI. GROUNDS OF REJECTION TO BE REVIEWED ON APPEAL

1. Claims 1-5, 11-17, 22-28, 34-37, 42-46, 48-52, 54, 49, 63-65, and 68 are rejected under 35 U.S.C. § 102(e) as being anticipated by Bittinger et al. (U.S. Patent 6,453,362).

2. Claims 18-20, 38-40, 47, 53 and 66 are rejected under 35 U.S.C. § 103(a) as being unpatentable over Bittinger in view of Leach (U.S. Patent 6,108,715).

3. Claims 21, 41, and 67 are rejected under 35 U.S.C. § 103(a) as being unpatentable over Bittinger in view of Instaweb Online Computing Dictionary (Instaweb, http://www.instaweb.com/foldoc.cgi?query=XML).

VII. ARGUMENT

First Ground of Rejection:

Claims 1-5, 11-17, 22-28, 34-37, 42-46, 48-52, 54, 49, 63-65, and 68 are rejected under 35 U.S.C. § 102(e) as being anticipated by Bittinger et al. (U.S. Patent 6,453,362). Appellants traverse this rejection for the following reasons. Different groups of claims are addressed under their respective subheadings.

Claims 1-4, 11-12, 16, 21-27, 41-46, 48-52, 54, 59 and 67-68:

Bittinger fails to disclose a method for remotely invoking methods in a distributed computing environment, comprising: a client generating a message in a data representation language, wherein the message includes information representing a computer programming language method call, and wherein the message further includes a credential for allowing the client access to a service configured to perform functions on behalf of clients in the distributed computing environment ... the service examining the credential included in the message; if said examining determines the credential is authentic, the service performing a function on behalf of the client in accordance with the information representing the computer programming language method call included in the message.

Bittinger teaches a method wherein a client invokes a remote server application using a server stub object that packs or marshals method parameters into a request (Bittinger, column 6, lines 9-17, and lines 28-33). Bittinger also discloses the use of systems such as CORBA, DCOM, RMI and Java to invoke methods on remote server applications. Note that CORBA, DCOM, RMI and Java are all code-centric technologies *that by definition do not employ data representation language messages for method calls.* Bittinger does not teach a client generating a message in a data representation language, wherein the message includes information representing a computer programming language method call. Instead, Bittinger teaches remote object communications using code-centric technologies such as CORBA, DCOM and RMI that employ programmatic method invocations.

In response to this argument, the Examiner states on p. 23 of the Final Action that "a data representation language seems to be nothing more than a method of encoding data for transmission between a client and a server where both the transmitter and receiver have means of reading the data encoded in the message." The Examiner's interpretation of generating a message in a data representation language representing a computer programming language method call is clearly incorrect and improper. By the express language of the claim, generating a message in a data representation language

representing a computer programming language method call requires more than just encoding data for transmission between a client and a server, and is distinctly different from remote method invocations using, for example, stubs, RMI or another code based communication technique. Note that the code-centric nature of prior art remote interfaces such as those used in Bittinger (e.g. CORBA and RMI) is discussed in Appellants' Related Art section on pp. 8-10 of the present application. In contrast, data representation languages are known in the art as languages that are used to represent or describe data in documents. The eXtensible Markup Language (XML) is one example of a data representation language. Therefore, "generating a message in a data representation language, wherein the message includes information representing a computer programming language method call" clearly means more than just any encoding of data. It requires the use of a data representation language. The term "language" has a specific meaning in the art. Furthermore, a data representation language is a specific type of language used for representing or describing data in documents. Although data representation languages, such as XML, are known in the prior art, they have not in the prior art generally been intended for use in programmatic interfaces such as the stub in Bittinger. In fact, such a use of a data representation language would be counter to the original intended purpose and normal use of data representation languages for representing or describing data in documents, as opposed to programmatic interfaces. Thus, Bittinger clearly does not anticipate generating a message in a data representation language representing a computer programming language method call.

In contrast, Bittinger describes a remote communication mechanism employing stubs and based on technologies such as CORBA or RMI. In the prior art, such mechanisms are specifically not based on data representation languages. A data representation language is a specific type of language having a specific purpose for describing data documents, as is well known to those of ordinary skill in the art. In the prior art, data representation languages were typically used to describe data documents or content (such as HTML or XML documents that may be displayed or processed in Internet communications). In Bittinger, clients use a programmatic (code-based) remote

interface to invoke services. The communications in Bittinger are clearly not data representation language messages. Furthermore, in Bittinger stubs are used to make an actual remote call. In other words, the client stub communications are actual CORBA or RMI code-centric method calls, not messages representing method calls.

Additionally, Bittinger fails to teach a data representation language message that includes information representing a computer programming language method call and a credential for allowing the client access to a service configured to perform functions on behalf of clients in the distributed computing environment. Bittinger teaches the use of a ticket object created by the client that a server application can use to pass a server stub object to the client. The client can then use the server stub object to invoke remote methods on the server application (Bittinger, column 6, lines 63-67, column 7, lines 42-47, and column 8, lines 15-22). The Examiner argues that Bittinger's login procedures and ticket object corresponds to a client including in a data representation language message a credential allowing the client access to a service. However, under Bittinger, the ticket is used to notify the client that the server application is running and ready to receive requests and provides a method for the server application to provide a server stub object to the client. Further, Bittinger teaches the use of a separate authentication server to authenticate the client using well-known login procedures such as user id and password. After authenticating the client, the authentication server then forwards the ticket to the server application (Bittinger, column 7, line 67 – column 8, line 8). The ticket in Bittinger is not in a message with a representation of a method call for a function of a service. Therefore, Bittinger does not teach the use of a ticket *as a credential* as suggested by the Examiner. Furthermore, the login procedure referred to by the Examiner in his Response to Arguments section has nothing to do with including a credential with a representation of a method call for a function of a service. Moreover, Bittinger clearly does not teach a single message that includes both a representation of a computer programming language method call and a credential for allowing the client access to a service.

Bittinger also fails to disclose the service examining the credential included in the message and performing a function on behalf of the client in accordance with the information representing the computer language method call included in the message if the credential is authentic. In contrast, Bittinger teaches the use of a separate server authenticating a client *prior to the launching* of the desired server application (Bittinger, column 8, lines 9-14). Bittinger fails to teach that any credential is verified by the service application that performs the function in accordance with a representation of the method call. Thus, Bittinger clearly does not disclose the service examining a credential included in the same message that includes a representation of the computer programming language method call.

The teachings of Bittinger clearly do not meet the standard for anticipation which requires that the identical invention must be shown in as complete detail as is contained in the claims. *Richardson v. Suzuki Motor Co.*, 868 F.2d 1226, 1236, 9 USPQ2d 1913, 1920 (Fed. Cir. 1989). Anticipation requires the presence in a single prior art reference disclosure of each and every element of the claimed invention, arranged as in the claim. *Lindemann Maschinenfabrik GmbH v. American Hoist & Derrick Co.*, 730 F.2d 1452, 1457, 221 USPQ 481, 485 (Fed. Cir. 1984). Appellants' invention as recited in claim 1 is clearly not anticipated by Bittinger.

Claims 5 and 28:

Regarding claim 5, the Examiner states that Bittinger discloses "the client message endpoint attaching the credential to the message (tStamp is an identifier used on all messages)." Appellants assert that the Examiner's interpretation of Bittinger is incorrect. The Examiner argues that Bittinger's tStamp equates to a credential attached to a data representation language message. However, Bittinger teaches, "the identifier 'tstamp' may be used *to locate the ticket* within the [client-side registry] database" (emphasis added, Bittinger, column 7, lines 8-9). Bittinger further describes a server using the tstamp to retrieve a ticket stub from the client-side registry (Bittinger, column 7, lines 41-43). After retrieving the ticket stub, the server invokes an acknowledgement

method of the ticket stub. Thus, Bittinger's tStamp cannot be considered any sort of credential. Nor does Bittinger teach that a tstamp is used on all messages, contrary to the Examiner's assertion. Bittinger's use of a tstamp to allow a server application to locate and retrieve a ticket stub from a client-side registry clearly does not constitute a client message endpoint attaching the credential to a data representation language message. Thus, claims 5 and 28 are clearly not anticipated by Bittinger.

Claims 13-14, 34-35 and 63-64:

In regard to claim 13, Bittinger does not teach a data representation language message including information representing the computer programming language method call that includes an identifier of the method call, and wherein said performing a function comprises regenerating the method call in accordance with the identifier of the method call included in the information representing the method call, and executing a computer programming language method of the service in accordance with the regenerated method call. In Bittinger, a client stub is used to make a remote call to a service using CORBA, RMI or a similar technology. As is well understood by anyone of ordinary skill in the art, in technologies such CORBA and RMI a method call is made remotely. CORBA and RMI doe not involve sending any representation of the method call that is then regenerated into the method call. Thus, Bittinger clearly does not anticipate claim 13.

Claims 15, 36 and 65:

Bittinger does not teach a method wherein a service includes a service method gate configured to provide an interface to the one or more computer programming language methods of the service by receiving data representation language messages and invoking computer programming language methods specified by the messages. The Examiner cites column 7, lines 41-57 of Bittinger and argues that Bittinger's server stub "is used as a gate to provide an interface to one or more computer programming language methods." However, the server stub in Bittinger does not provide an interface to computer programming language methods of a service by receiving data representation

language messages. Instead, the server stub in Bittinger allows a client to *invoke* methods of the server stub. The purpose of Bittinger's server stub (as well as all RMI stub objects) is to allow a client to interact with the server stub as if it were the remote server object. Nowhere does Bittinger mention a server stub receiving data representation language messages, and, as described above, Bittinger's remote interfaces (such as RMI and CORBA) do not (by definition) use data representation language messages.

Claims 17 and 37:

Bittinger fails to teach a service message endpoint configured to send messages in the data representation language to the client for the service. The Examiner cites a completely irrelevant passage (column 8, lines 32-45) where Bittinger describes how the steps illustrated in FIG. 4 can be implemented by computer program instructions. FIG. 4 of Bittinger does not mention anything about a service message endpoint configured to send a message in a data representation language. Instead, FIG. 4 only illustrates a method for a client to login and obtain a server stub object. Bittinger only teaches that a client uses a server stub object to make requests to the server application (column 8, lines 29-31) but does not describe any sort of server message endpoint configured to send messages in the data representation language to the client for the service. Furthermore, Bittinger's use of prior art remote interfaces, such as CORBA and RMI, actually teaches away from a service message endpoint sending messages in a data representation language, as it is well-known that such remote interface do not use data representation language messages. Thus, Bittinger fails to anticipate claims 17 and 37.

Second Ground of Rejection:

Claims 18-20, 38-40, 47, 53 and 66 are rejected under 35 U.S.C. § 103(a) as being unpatentable over Bittinger in view of Leach (U.S. Patent 6,108,715). Appellants traverse this rejection for the following reasons. Different groups of claims are addressed under their respective subheadings.

Claims 18, 38, 47, 53 and 66:

In regard to claim 18, Bittinger in view of Leach does not teach or suggest storing the generated results data to a space service in the distributed computing environment, providing an advertisement for the stored results data to the client, wherein the advertisement comprises information to enable access by the client to the stored results data, and the client accessing the stored results data from the space service in accordance with the information in the provided advertisement. The Examiner states, "Leach discloses the creation of a data stack used to store the results of the operations locally on the server, then allows the client to map back to the stored results stack by providing an address which allows direct transfer between the client and the server greatly reducing the processing overhead." Appellants fail to see the relevance of Leach's teachings to the limitations of claim 18. Furthermore, Appellants respectfully disagree with the Examiner's interpretation of Leach.

Leach teaches direct stack-to-stack transfer of parameters between two processes executing concurrently on a single machine. Rather than marshaling and unmarshaling parameters when invoking a remote procedure call in a second process, Leach teaches the use of a shared data stack and that a operating system kernel process copies the input and output parameters between the two processes individual stacks. (Leach, column 4, lines 32-43). Further, Leach teaches directly copying data from a server process to a client process through the use of a kernel that can access the address spaces of each process, thereby facilitating such a direct transfer of data (Leach, column 5, lines 44-52). Thus, contrary to the Examiner's assertion, Leach does not disclose the creation of a data stack used to store the results of the operations locally on the server and then allows the client to map back to the stored results stack by providing an address.

Specifically, under Leach, the client is not provided an address to map back to the any stored results stack, as the Examiner contends. Instead, Leach teaches that a kernel process directly transfers the output parameters onto the client's stack and "transfers

control to the client process with the instruction pointer pointing to the instruction that follows the client process' call to the proxy method" (Leach, column 5, lines 52-55). Hence, Leach's kernel transfers any results directly into the client's stack and does not provide the client any address or information that would allow the client to "map back" to the stored results stack. Further, Leach teaches that the kernel must do this, because the client would not be able to access the other process' stack (Leach, column 6, lines 51-54).

Bittinger in view of Leach clearly does no teach or suggest storing the generated results data to a space service in the distributed computing environment, providing an advertisement for the stored results data to the client, wherein the advertisement comprises information to enable access by the client to the stored results data, and the client accessing the stored results data from the space service in accordance with the information in the provided advertisement. Instead, the combination of Bittinger and Leach proposed by the Examiner would result in a system in which a client uses Bittinger's server stub to invoke methods of a remote server application and wherein any results from the method would be transferred to the client's stack as taught by Leach.

Claims 19 and 39:

Bittinger in view of Leach fails to teach or suggest generating a client results message endpoint in accordance with the information in the provided advertisement, wherein the client results message endpoint is configured to send messages in the data representation language to the space service for the client; generating a results request message in the data representation language, wherein the results request message requests the results data be provided to the client; the client results message endpoint sending the results request message to the space service; and the space service sending the requested results data to the client results message endpoint in response to the results request message. The Examiner has failed cite any portion of Bittinger or Leach that mentions anything about a space service or about the generation and use of either a client results message endpoint or results request messages. The Examiner cites only Leach's teachings regarding a kernel that copies output data resulting from a remote procedure

call directly from the remote procedure's stack to the client's stack. Leach teaches however that the kernel copies the output data after capturing processing control when the remote procedure has completed its execution (Leach, column 3, lines 39-41).

Leach does not mention anything about generating a client results message endpoint, nor about generating a results request message in a data representation language. In fact, Leach teaches away from generating a client results message endpoint and generating a results request message in a data representation language by teaching the automatic and direct transfer of output data from the remote procedure's stack to the client stack. Once the output data is in the client's stack, the client has direct and complete access to them. There is no need, nor does it make sense, to generate a results message endpoint or to generate results request messages to obtain results data when the results data are already readily available in the client stack.

Claims 20 and 40:

Bittinger in view of Leach fails to teach or suggest wherein the information (in the advertisement for the stored results data) to enable access by the client to the stored results data comprises one or more Uniform Resource Identifiers (URIs) for accessing the stored results data. The Examiner contends, without citing any passage of Leach, that Leach teaches the use of an address for access a data stack and that "the types of files are known by the client, which are key features of a URI". The Examiner is presumably referring to Leaches teachings regarding directly transferring results data from a remote procedures address space to a clients stack. However, Leach teaches nothing regarding Uniform Resource Identifiers or anything that could be considered a URI. The only addresses Leach refers to are memory address in virtual memory address spaces within a computer system. Furthermore, Leach does not mention anything regarding types of files that are known by a client, as the Examiner asserts. As described above, Leach teaches a method for automatically and directly transferring output data from a remote procedure's address space to a client's stack, thereby giving the client ready and easy access to the output data. Leach fails to teach anything regarding an advertisement for stored results

data and further fails to disclose that such an advertisement comprises information to enable access by the client to the stored results data that includes one or more URIs. Additionally, there is no need for a client to use URIs to access stored results data because Leach copies output data to the client's stack, thereby providing the client direct access to the output data. Furthermore, one of ordinary skill in the art would not use a URI for a client to access data stored on the client's own stack as in the system of Leach.

Third Ground of Rejection:

Claims 21, 41 and 67 are rejected under 35 U.S.C. § 103(a) as being unpatentable over Bittinger in view of InstaWeb. Appellants traverse this rejection for the following reasons.

Claims 21, 41 and 67 are patentable for at least the reasons given above regarding their respective independent claims.

Additionally, regarding claim 21, the Examiner states, "XML can be used to create custom tags for data objects that offer greater flexibility in organizing and presenting information." However, Appellants' claim 21 is not concerned with data objects or in organizing or presenting information. Neither is Bittinger. Bittinger teaches a method for remote invocation of server applications. Specifically, Bittinger teaches a client login and registration process to coordinate the launching of a server application and relies upon well known, prior-art, remote method invocations between clients and server applications (*See*, Bittinger, column 5, line 55-column 6, line 28).

Furthermore, the combination of Bittinger and InstaWeb would not result the method recited by Appellants' claim 21. InstaWeb teaches only that XML can be used to create custom tags for data objects that offer greater flexibility in organizing and presenting information. InstaWeb does not suggest that XML be used for generating messages that include information representing computer programming language method calls or that include credentials for allowing a client access to a service configured to

perform functions on behalf of clients, as recited in Appellants' claim 21. Bittinger relies upon remote invocation methods such as RMI and CORBA that do not use data representation languages. Any combination of Bittinger and InstaWeb would still use prior-art remote invocations (which do not use data representation language messaging), as taught by Bittinger. Thus, the Examiner's proposed combination would result only in a system that uses Bittinger's system for launching a server application, but that also uses XML for organizing and presenting information, as taught by InstaWeb.

VIII. CONCLUSION

For the foregoing reasons, it is submitted that the Examiner's rejection of claims 1-5, 11-28, 34-54, 59 and 63-68 was erroneous, and reversal of his decision is respectfully requested.

The Commissioner is authorized to charge the appeal brief fee of $340.00 and any other fees that may be due to Meyertons, Hood, Kivlin, Kowert, & Goetzel, P.C. Deposit Account No. 501505/5181-67300/RCK. This Appeal Brief is submitted with a return receipt postcard.

Respectfully submitted,

Robert C. Kowert
Reg. No. 39,255
Attorney for Appellants

Meyertons, Hood, Kivlin, Kowert & Goetzel, P.C.
P.O. Box 398
Austin, TX 78767-0398
(512) 853-8850

Date: December 1, 2004

IX. CLAIMS APPENDIX

The claims on appeal are as follows.

1. A method for remotely invoking methods in a distributed computing environment, comprising:

a client generating a message in a data representation language, wherein the message includes information representing a computer programming language method call, and wherein the message further includes a credential for allowing the client access to a service configured to perform functions on behalf of clients in the distributed computing environment;

the client sending the message to the service;

the service examining the credential included in the message;

if said examining determines the credential is authentic, the service performing a function on behalf of the client in accordance with the information representing the computer programming language method call included in the message; and

if said examining determines the credential is not authentic, the service not performing the function on behalf of the client.

2. The method as recited in claim 1, wherein the client comprises a client method gate configured to provide an interface to the service by generating data representation language messages including information representing method calls, and wherein said generating a message is performed by the client method gate.

3. The method as recited in claim 2, wherein said sending the message is performed by the client method gate.

4. The method as recited in claim 2, wherein the client further comprises a client process, the method further comprising:

the client process generating the computer programming language method call; and

the client method gate receiving the method call generated by the client process;

wherein said generating a message is performed in response to said receiving the method call.

5. The method as recited in claim 2, wherein the client further comprises a client message endpoint, wherein said sending the message to the service comprises:

the client method gate sending the message to the client message endpoint, wherein the client message endpoint is configured to send messages in the data representation language to the service;

the client message endpoint attaching the credential to the message; and

the client message endpoint sending the message to the service.

11. The method as recited in claim 1, wherein the service comprises a service message endpoint configured to receive messages in the data representation language from the client, wherein said performing a function comprises the service message endpoint receiving the message from the client.

12. The method as recited in claim 1, wherein the service comprises one or more computer programming language methods executable within the service, wherein said performing a function comprises executing a computer programming language method of the service in accordance with the information representing the computer programming language method call included in the message.

13. The method as recited in claim 1, wherein the service comprises one or more computer programming language methods executable within the service, wherein the information representing the computer programming language method call includes an identifier of the method call, and wherein said performing a function comprises:

regenerating the method call in accordance with the identifier of the method call included in the information representing the method call; and

executing a computer programming language method of the service in accordance with the regenerated method call.

14. The method as recited in claim 13, wherein the information representing the computer programming language method call further includes one or more parameter values of the method call, and wherein said executing a computer programming language method in accordance with the regenerated method call comprises providing the one or more parameter values from the information representing the method call as parameter values of the method call.

15. The method as recited in claim 13, wherein the service further comprises a service method gate configured to provide an interface to the one or more computer programming language methods of the service by receiving data representation language messages and invoking computer programming language methods specified by the messages, and wherein said regenerating the method call is performed by the service method gate.

16. The method as recited in claim 1, wherein said performing a function generates results data, the method further comprising the service providing the generated results data to the client.

17. The method as recited in claim 1, wherein said performing a function generates results data, and wherein the service comprises a service message endpoint configured to send messages in the data representation language to the client for the service, the method further comprising:

the service message endpoint sending a results message to the client, wherein the results message includes the generated results data.

18. The method as recited in claim 1, wherein said performing a function generates results data, the method further comprising:

storing the generated results data to a space service in the distributed computing environment;

providing an advertisement for the stored results data to the client, wherein the advertisement comprises information to enable access by the client to the stored results data; and

the client accessing the stored results data from the space service in accordance with the information in the provided advertisement.

19. The method as recited in claim 18, wherein the client accessing the stored results data comprises:

generating a client results message endpoint in accordance with the information in the provided advertisement, wherein the client results message endpoint is configured to send messages in the data representation language to the space service for the client;

generating a results request message in the data representation language, wherein the results request message requests the results data be provided to the client;

the client results message endpoint sending the results request message to the space service; and

the space service sending the requested results data to the client results message endpoint in response to the results request message.

20. The method as recited in claim 18, wherein the information to enable access by the client to the stored results data comprises one or more Uniform Resource Identifiers (URIs) for accessing the stored results data.

21. The method as recited in claim 1, wherein said data representation language is eXtensible Markup Language (XML).

22. The method as recited in claim 1, wherein said computer programming language is the Java programming language, and wherein the information representing the method call in the message represents a Java method call to a Java method implemented on the service, and wherein the service performing a function comprises invoking the Java method on the service in accordance with the information representing the Java method call included in the message.

23. The method as recited in claim 1, wherein the client is executing within a virtual machine, wherein the virtual machine is executing within a client device in the distributed computing environment.

24. The method as recited in claim 16, wherein the virtual machine is a Java Virtual Machine (JVM).

25. A distributed computing system comprising:

a service device comprising one or more functions executable on the service device on behalf of client devices in the distributed computing system;

a client device configured to:

generate a message in a data representation language, wherein the message includes information representing a computer programming language method call, and wherein the message further includes a credential for allowing the client device access to the service device; and

send the message to the service device;

wherein the service device is configured to:

examine the credential included in the message;

if said examining verifies the credential, perform a function on behalf of the client in accordance with the information representing the computer programming language method call included in the message; and

if said examining does not verify the credential, not perform the function on behalf of the client.

26. The system as recited in claim 25, wherein the client device comprises a client method gate configured to provide an interface to the service by generating data representation language messages including information representing method calls, and wherein said generating a message is performed by the client method gate.

27. The system as recited in claim 26, wherein the client device further comprises a client process,

wherein the client process is configured to generate the computer programming language method call;

wherein the client method gate is further configured to receive the method call generated by the client process; and

wherein said generating a message is performed by the client method gate in response to said receiving the method call.

28. The system as recited in claim 26, wherein the client device further comprises a client message endpoint,

wherein the client method gate is further configured to send the message to the client message endpoint; and

wherein the client message endpoint is configured to:

attach the credential to the message; and

send the message to the service device.

34. The system as recited in claim 25, wherein the service device comprises one or

more computer programming language methods executable within the service device, wherein the information representing the computer programming language method call includes an identifier of the method call, and wherein, in said performing a function, the service device is further configured to:

regenerate the method call in accordance with the identifier of the method call included in the information representing the method call; and

execute a computer programming language method of the service device in accordance with the regenerated method call.

35. The system as recited in claim 34, wherein the information representing the computer programming language method call further includes one or more parameter values of the method call, and wherein, in said executing a computer programming language method in accordance with the regenerated method call, the service device is further configured to:

provide the one or more parameter values from the information representing the method call as parameter values of the method call.

36. The system as recited in claim 34, wherein the service device further comprises a service method gate configured to provide an interface to the one or more computer programming language methods of the service by receiving data representation language messages and invoking methods specified by the messages, and wherein said regenerating the method call is performed by the service method gate.

37. The system as recited in claim 25, wherein said performing a function generates results data, wherein the service device comprises a service message endpoint configured to send a results message in the data representation language to the client device, wherein the results message includes the generated results data.

38. The system as recited in claim 25, further comprising:

a space service;

wherein the service device is further configured to:

store results data generated by said performing a function to the space service;

provide an advertisement for the stored results data to the client device, wherein the advertisement comprises information to enable access by the client device to the stored results data; and

wherein the client device is further configured to access the stored results data from the space service in accordance with the information in the provided advertisement.

39. The system as recited in claim 38,

wherein, in accessing the stored results data, the client device is further configured to generate a client results message endpoint in accordance with the information in the provided advertisement;

wherein the client results message endpoint is configured to:

generate a results request message in the data representation language, wherein the results request message requests the results data be provided to the client device; and

send the results request message to the space service; and

wherein the space service is configured to send the requested results data to the client results message endpoint in response to the results request message.

40. The system as recited in claim 38, wherein the information to enable access by the client device to the stored results data comprises one or more Uniform Resource Identifiers (URIs) for accessing the stored results data.

41. The system as recited in claim 25, wherein said data representation language is eXtensible Markup Language (XML).

42. The system as recited in claim 25, wherein said computer programming language is the Java programming language, and wherein the information representing the method call in the message represents a Java method call to a Java method implemented on the service, and wherein, in said performing a function, the service device is further configured to invoke the Java method on the service device in accordance with the information representing the Java method call included in the message.

43. The system as recited in claim 25, further comprising:

a virtual machine executable within the client device; and

a client process executable within the virtual machine, wherein said generating a message and said sending the message are performed by the client process.

44. The system as recited in claim 43, wherein the virtual machine is a Java Virtual Machine (JVM).

45. A device comprising:

a client component; and

a method gate;

wherein the client component is configured to generate a computer programming language method call;

wherein the method gate is configured to:

access the computer programming language method call generated by the client component;

generate a message in a data representation language, wherein the message includes information representing a computer programming language method call, and wherein the message further includes a credential for allowing the client device access to a service in a distributed computing environment; and

send the message to the service;

wherein the service is operable to verify the message as authentic by examining the credential included in the message, and to perform a function on behalf of the client component in accordance with the information representing the computer programming language method call included in the message if the message is verified as authentic.

46. The device as recited in claim 45, wherein the method gate comprises a data representation language message schema comprising descriptions of data representation language messages the device is authorized to send to the service, wherein said generating a message is performed in accordance with a description of the message comprised in the message schema.

47. The device as recited in claim 45, wherein the service is further operable to store results data generated by the function to a space service in the distributed computing environment, and wherein the client component is further configured to:

access a data representation language advertisement for the results data, wherein the advertisement comprises information to enable access by the client component to the results data; and

access the results data from the space service in accordance with the information in the provided advertisement for the stored results data.

48. The device as recited in claim 45, wherein said computer programming language is the Java programming language, and wherein the information representing a method call in the message represents a Java method call to a Java method implemented on the service.

49. A device comprising:

a client component configured to generate a message in a data representation language, wherein the message includes information representing a computer programming language method call; and

a message endpoint configured to:

attach a credential to the message for allowing the client component access to a service in a distributed computing environment; and

send the message to a service in a distributed computing environment;

wherein the service is operable to verify the message as authentic by examining the credential included in the message, and to perform a function on behalf of the client component in accordance with the information representing the computer programming language method call included in the message if the message is authentic.

50. The device as recited in claim 49, wherein the client component is further configured to generate the computer programming language method call, and wherein said generating a message is performed in response to said generating the computer programming language method call.

51. The device as recited in claim 49, wherein the device further comprises a virtual machine executable within the device, wherein the client component and the message endpoint are executable within the virtual machine.

52. The device as recited in claim 51, wherein the virtual machine is a Java Virtual Machine (JVM).

53. The device as recited in claim 49, wherein the service is further operable to store results data generated by the function to a space service in the distributed computing environment, and wherein the client component is further configured to:

access a data representation language advertisement for the results data, wherein the advertisement comprises information to enable access by the client component to the results data; and

access the results data from the space service in accordance with the information in the provided advertisement for the stored results data.

54. The device as recited in claim 49, wherein said computer programming language is the Java programming language, and wherein the information representing a method call in the message represents a Java method call to a Java method implemented on the service.

59. A carrier medium comprising program instructions, wherein the program instructions are computer-executable to implement:

a client generating a message in a data representation language, wherein the message includes information representing a computer programming language method call, and wherein the message further includes a credential for allowing the client access to a service configured to perform functions on behalf of clients in the distributed computing environment;

the client sending the message to the service;

the service examining the credential included in the message;

if said examining determines the credential is authentic, the service performing a function on behalf of the client in accordance with the information representing the computer programming language method call included in the message; and

if said examining determines the credential is not authentic, the service not performing the function on behalf of the client.

63. The carrier medium as recited in claim 59, wherein the service comprises one or more computer programming language methods executable within the service, wherein the information representing the computer programming language method call includes an identifier of the method call, and wherein, in said performing a function, the program instructions are further computer-executable to implement:

regenerating the method call in accordance with the identifier of the method call included in the information representing the method call; and

executing a computer programming language method of the service in accordance with the regenerated method call.

64. The carrier medium as recited in claim 63, wherein the information representing the computer programming language method call further includes one or more parameter values of the method call, and wherein, in said executing a computer programming language method in accordance with the regenerated method call, the program instructions are further computer-executable to implement providing the one or more parameter values from the information representing the method call as parameter values of the method call.

65. The carrier medium as recited in claim 63, wherein the service further comprises a service method gate configured to provide an interface to the one or more computer programming language methods of the service by receiving data representation language messages and invoking computer programming language methods specified by the messages, and wherein said regenerating the method call is performed by the service method gate.

66. The carrier medium as recited in claim 59, wherein said performing a function generates results data, and wherein the program instructions are further computer-executable to implement:

storing the generated results data to a space service in the distributed computing environment;

providing an advertisement for the stored results data to the client, wherein the advertisement comprises information to enable access by the client to the stored results data; and

the client accessing the stored results data from the space service in accordance with the information in the provided advertisement.

67. The carrier medium as recited in claim 59, wherein said data representation language is eXtensible Markup Language (XML).

68. The carrier medium as recited in claim 59, wherein said computer programming language is the Java programming language, and wherein the information representing the method call in the message represents a Java method call to a Java method implemented on the service, and wherein, in said performing a function, the program instructions are further computer-executable to implement invoking the Java method on the service in accordance with the information representing the Java method call included in the message.

X. EVIDENCE APPENDIX

No evidence submitted under 37 CFR §§ 1.130, 1.131 or 1.132 or otherwise entered by the Examiner is relied upon in this appeal.

XI. RELATED PROCEEDINGS APPENDIX

There are no related proceedings.